ISO 29990:2010

NATIONAL INFOCOMM COMPETENCY FRAMEWORK

NUS National University of Singapore

ISS INSTITUTE OF SYSTEMS SCIENCE

# FUNDAMENTAL OF PROGRAMMING IN C#

## STRING HANDLING

# Objectives

- Write a program that manipulates string for various purpose
- Write a program that is able to manipulate string by its characters

# Agenda

- String Operation
- String Function
- String Comparisons
- Case Examples

# Strings Revisited

- Strings are series of characters.
- Strings Constant never change and are indicated by double quotes.
  - Examples: “Institute of Systems Science” “monkey”
- Variable strings are a special type of variable that are capable of storing strings.
  - Declaration:

```
string Country;
```

# String Operations

- Strings like numerals are capable of being manipulated or operated upon. The following are the types of operations that you can perform on Strings:
  - Operators for putting strings together.
  - Functions for pulling strings apart.
  - Functions for creating strings.
  - Functions for modifying strings.
  - Operators for finding out things about strings.

Note: In any of the above operators and functions you can use either a variable or constant string.

# String Operations

- Strings like numerals are capable of being manipulated or operated upon.

- There are many ways by which you can perform an operation on a string in C#
  - Use of Operators
  - Use of methods by treating "string" as an object
  - Use of static methods in the String class

- We would confine to the first two approaches in this lesson.

# String Operations

- Use of operator on a string.
  - Comparison
    - Two strings can be compared with the help of relational operators just the way we do it for the numeric.
    - Example

      Is ("Monkey" == "Donkey") ? Answer is NO.

      Is ("Monkey" != "Donkey") ? Answer is YES

      Is ("Monkey" == "monkey") ? Answer is NO
    - The above Boolean (relational) expressions is usually used in "if" statements or while/for/select statements.
    - You cannot use the other relational operators like > and < on the strings.
    - Example

      Is ("Monkey" > "Donkey") ? This will generate an error!

# String Operations

- Use of operator on a string.
  - Concatenation
    - Strings are concatenated (joined together) using a + operator.
    - Example:

      “Venkat” + “Raman” => “VenkatRaman”

      “Hickory “ + “Dickory ” + “Dock” => “Hickory Dickory Dock”

# String Properties

- Length
  - The Length property of the string provides the length of the string object
  - Length is read only property
  - Example:

Program:

```
string s = "venkat";
Console.WriteLine(s.Length);
```

Output:

```
6
```

# String Methods

- CompareTo
  - The CompareTo method of the string takes another string as argument.
  - The return value is:
    - 0 if the string equals the argument
    - 1 if the string is greater than the argument
    - -1 if the string is less than the argument

  Example:

  Program:

```
string s = "abc";
string r = "xyz"
Console.WriteLine(s.CompareTo(r));
```

  Output:

```
-1
```

# String Methods

- Trim
  - The Trim method if used without an argument, removes white spaces at the beginning and end of the string.

  Example:

Program:

```
string s = "   abc   ";
Console.WriteLine("*" + s + "*");
string r = s.Trim()
Console.WriteLine ("*" + r + "*");
```

Output:

```
*   abc   *
*abc*
```

# String Methods

- Trim(chars)
  - The Trim method if used without an argument array of characters, removes specified Unicode characters at the beginning and end of the string.

  Example:

Program:

```
string s = "$%$$abc%s%$";
char[] c = new char[] {'$','%'};
Console.WriteLine("*" + s + "*");
string r = s.Trim(c);
Console.WriteLine ("*" + r + "*");
```

Output:

```
*$%$$abc%s%$*
*abc%s*
```

# String Methods

- TrimEnd(chars)
- TrimStart(chars)
  - The above two methods work the same way as the Trim method with a character array argument. The only difference is:
    - TrimEnd trims the specified characters at the end of the string only leaving the head of the string untouched
    - TrimStart trims the specified characters at the beginning of the string only leaving the tail untouched

# String Methods

- Substring(int,int)
  - Substring method of the string extracts a part of the string
  - The Substring method uses two integer arguments. The first argument indicates the starting position and second argument the length

  Example:

Program:

```
string s = "Venkatraman";
string r = s.Substring(4,3);
Console.WriteLine(r);
Console.WriteLine (s.Substring(6,4));
```

Output:

```
atr
rama
```

# String Methods

- Equals(string)
  - Equals method compares the string with the argument and returns a true if the two strings are same else it returns a false.

Example:

Program:

```
string s = "abc";
string r = "xyz";
if ( s.Equals(r) )
   Console.WriteLine( "s and r are same");
else
   Console.WriteLine( "s and r are not same");
if (s.Equals("abc"))
    Console.WriteLine(" s is \"abc\" ");
```

Output:

```
s and r are not same
s is "abc"
```

# String Methods

- Insert(int, string)
  - Insert method inserts into the string another string (given as second argument) from position given in the first argument and returns the new value.

Example:

Program:

```
string s = "Institute Systems Science";
string r = s.Insert(10, "of ");
Console.WriteLine(s);
Console.WriteLine(r);
```

Output:

```
Institute Systems Science
Institute of System Science
```

# String Methods

- PadLeft(int n, char c)
- PadRight(int n, char c)
  - PadLeft pads n number of characters 'c' to the left of the string.
  - PadRight does the same on the right side.

Example:

Program:

```
string s = "ABC";
string r = s.PadLeft(7, 'c');
Console.WriteLine(s);
Console.WriteLine(r);
Console.WriteLine(s.PadRight(6,'z'));
```

Output:

```
ABC
ccccABC
ABCzzz
```

# String Methods

- ToUpper()
- ToLower()
  - ToUpper method returns the string value in Upper Case
  - ToLower method returns the string value in Lower Case

  Example:

Program:

```
string s = "Venkat";
string r = s.ToUpper();
Console.WriteLine(s);
Console.WriteLine(r);
Console.WriteLine(s.ToLower());
```

Output:

```
Venkat
VENKAT
venkat
```

# String and Characters

- Characters that makes up a string can be read individually using the index
  - The index starts with 0. The first character of the string has the index of 0
  - But they cannot be modified

```
string s="Hello";
Console.WriteLine(s[1]);

s[1] = 'b';
```

```
e

Error!
```

  - If you want to be able to modify each of the character, you can check StringBuffer class

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/strings/

# String and Characters

- string is a data type that store string/sequence of zero to many characters. E.g.: "", "a", "abc"
- char is a data type that store ONE character.
  - E.g.: 'a' or 'b'
  - These are not valid: '' and 'ab'
- char is written with a single quote '
- If `s` is a string, `s[3]` returns similar value as `s.Substring(3,1)` but different data types.

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/char

# Interesting facts about char

- `char` can be implicitly converted into `int`
  - This is inherited from the fact that in C language, char is also treated as number
  - But the reverse is not true.

```
char x = 'a';
Console.WriteLine(x);
int y = x;
Console.WriteLine(y);
Console.WriteLine((char)y);
Console.WriteLine('9' - '0');
Console.WriteLine('z' - 'a');
```

```
a

97
a
9
25
```

# Summary

- string is an object that has useful property like Length and methods that can be used to manipulate strings
- string.Split() is one of the most useful function for parsing a string
  - Breaking a string into words
  - Read a comma separated values (CSV)
- Individual characters in a string can be individually addressed
- Note some differences between string and char.
  - String is more flexible and easier to use